In [254]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
```

In [255]:

```
heart = pd.read_csv("E:\\Datasets\\DataSet\\train_2v.csv")
```

In [256]:

```
heart.head()
```

Out[256]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_
0	30669	Male	3.0	0	0	No	children	
1	30468	Male	58.0	1	0	Yes	Private	
2	16523	Female	8.0	0	0	No	Private	
3	56543	Female	70.0	0	0	Yes	Private	
4	46136	Male	14.0	0	0	No	Never_worked	

In [272]:

```
count=heart['stroke'].value_counts()
count1 = heart['Residence_type'].value_counts()
print(count)
print(count1)
```

```
0    42617
1      783
Name: stroke, dtype: int64
Urban    21756
Rural    21644
Name: Residence_type, dtype: int64
```

In [290]:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
lencoder = LabelEncoder()
heart.iloc[:, 1:2] = lencoder.fit_transform(heart.iloc[:, 1:2])
heart.iloc[:, 5:6] = lencoder.fit_transform(heart.iloc[:, 5:6])
heart.iloc[:, 7:8] = lencoder.fit_transform(heart.iloc[:, 7:8])
```

In [291]:

```
from sklearn.preprocessing import Imputer
imputer_mean = Imputer(missing_values='NaN', strategy="mean", axis=0)
imputer_most_frequent = Imputer(missing_values='NaN', strategy="most_frequent", axis=0)
heart.iloc[:, 9:10]= imputer_mean.fit_transform(heart.iloc[:, 9:10])
```

```
C:\Users\Personal\Anaconda\lib\site-packages\sklearn\utils\deprecation.py:
66: DeprecationWarning: Class Imputer is deprecated; Imputer was deprecate
d in version 0.20 and will be removed in 0.22. Import impute.SimpleImputer
from sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\Personal\Anaconda\lib\site-packages\sklearn\utils\deprecation.py:
66: DeprecationWarning: Class Imputer is deprecated; Imputer was deprecate
d in version 0.20 and will be removed in 0.22. Import impute.SimpleImputer
from sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
```

In [274]:

```
x= heart.iloc[:, [1,2,3,4,5,7]]
y= heart.iloc[:, 11]
```

In [275]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=0)
```

Do if needed , "Balancing the data" Coding

from imblearn.over_sampling import RandomOverSampler

balance = RandomOverSampler(ratio=0.5)

x,y= balance.fit_sample(x,y)

In [276]:

```
print(x.shape)
print(y.shape)
print(heart.shape)
```

```
(43400, 6)
(43400,)
(43400, 12)
```

Logistic regression

In [323]:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(x_train,y_train)
```

```
C:\Users\Personal\Anaconda\lib\site-packages\sklearn\linear_model\logisti
c.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)
```

Out[323]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=
0,
                   warm_start=False)
```

In [324]:

```
stroke=classifier.predict(x_test)
stroke
```

Out[324]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [325]:

```
aa=classifier.score(x_train,y_train)
aa
```

Out[325]:

```
0.981500987491771
```

In [326]:

```
bb=classifier.score(x_test,y_test)
bb
```

Out[326]:

```
0.983026113671275
```

In [327]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
confusion_matrix(y_test,stroke)
```

Out[327]:

```
array([[12799,     0],
       [  221,     0]], dtype=int64)
```

In [328]:

```
accuracy_score(y_test,stroke)
```

Out[328]:

```
0.983026113671275
```

SVC

In [294]:

```
from sklearn.svm import SVC
classifier = SVC(kernel='linear')
classifier.fit(x_train,y_train)
```

Out[294]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

In [295]:

```
stroke=classifier.predict(x_test)
stroke
```

Out[295]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [296]:

```
aa=classifier.score(x_train,y_train)
aa
```

Out[296]:

```
0.981500987491771
```

In [297]:

```
bb=classifier.score(x_test,y_test)
bb
```

Out[297]:

```
0.983026113671275
```

In [298]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
confusion_matrix(y_test,stroke)
```

Out[298]:

```
array([[12799,     0],
       [  221,     0]], dtype=int64)
```

Random Forest Classifier

In [312]:

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier()
classifier.fit(x_train,y_train)
```

```
C:\Users\Personal\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:24
5: FutureWarning: The default value of n_estimators will change from 10 in
version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[312]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gin
i',
                       max_depth=None, max_features='auto', max_leaf_nodes
=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [313]:

```
stroke=classifier.predict(x_test)
stroke
```

Out[313]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [314]:

```
aa=classifier.score(x_train,y_train)
aa
```

Out[314]:

```
0.9820934825543121
```

In [315]:

```
bb=classifier.score(x_test,y_test)
bb
```

Out[315]:

```
0.9823348694316436
```

In [316]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
confusion_matrix(y_test,stroke)
```

Out[316]:

```
array([[12788,    11],
       [  219,     2]], dtype=int64)
```